



Microprocesadores

Capitulo 2

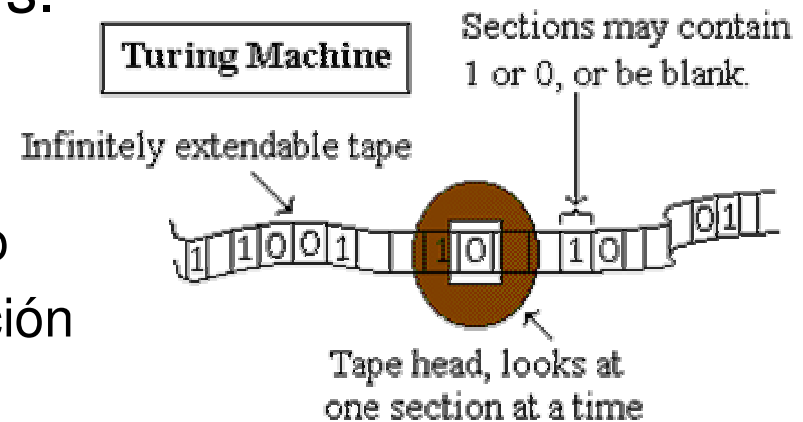


Capítulo 2

- Temario
 - Maquina de Turing
 - La Arquitectura de Von Neumann
 - Microprocesadores
 - Programas y Ejecución Instrucciones
 - Direccionamiento
 - Maquinas Virtuales
 - Assembler
 - Arquitecturas RISC & CISC
 - Arquitectura INTEL
 - Técnicas para mejorar rendimiento

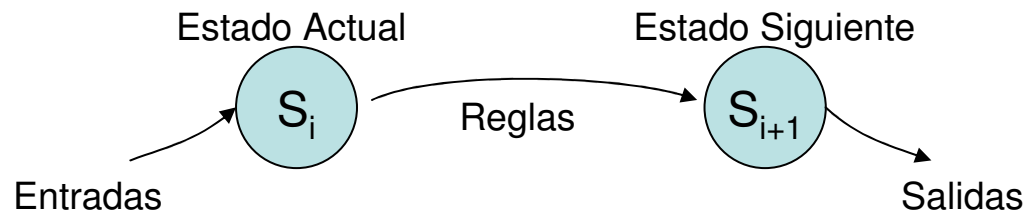
La máquina de Turing

- Alan Turing
 - Maquina de Turing (1936)
 - Paper: "*Computing machinery and intelligence*" (1950)
 - La maquina puede ser vista como un lector de cinta, que solo puede leer 1 posición de una cinta, que contiene blancos, 0s o 1s.
 - La maquina solo puede:
 - Escribir 0, 1 o Blanco
 - Cambiar el estado interno
 - Mover la cinta 0 o 1 posición (derecha/izquierda)



La máquina de Turing

- Máquina de Estados Finitos (o Autómata Finito)
 - La maquina tiene (en cualquier instante de tiempo discreto) un Estado Interno (S), el cual puede modificar en base a entradas que recibe desde el exterior, así como producir salidas.



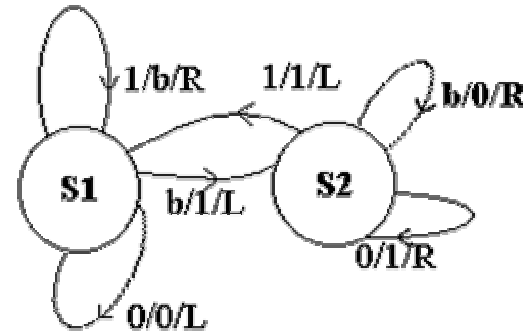
- Si tuviera 2 estados S1 y S2, distintas entradas, producirían salidas (movimientos y escrituras), y un cambio de estado

State	Read	Write	Move	Next State
S1	0	0	L	S1
	blank	1	L	S2
	1	blank	R	S1
S2	0	1	R	S2
	blank	0	R	S2
	1	1	L	S1

State Transition Table for a Turing Machine

La máquina de Turing

- Diagrama de Estado
 - El comportamiento de una maquina de Turing se puede representar en un diagrama de estado.
 - Para el ejemplo anterior:



Transition State Diagram of Turing Machine (corresponds to transition state table)



La máquina de Turing

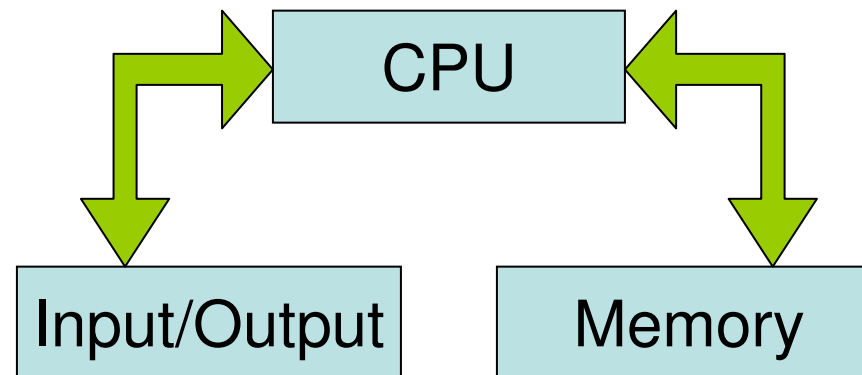
- El poder la Maquina de Turing
 - La “cinta” contiene:
 - Los datos de entrada de la maquina
 - Los datos intermedios (borrador)
 - Los datos de salida de la maquina
- } Memoria?
- El poder de la maquina se basa en la información que tenga en la “cinta” (o el largo de la cinta)
 - Con la información adecuada en la “cinta” la maquina de Turing puede emular acciones de distintas maquinas.
 - Incluso Turing planteaba que se podría construir una maquina “Universal”, que tomara las reglas de la propia cinta.

La Arquitectura de Von Neumann

- La Arquitectura de Von Neumann (1946)
 - Existe una CPU – Central Processing Unit.
 - Instrucciones y Datos son guardados en la misma memoria.
 - La CPU accede a la memoria, lee instrucciones, las decodifica y las ejecuta.
 - La CPU interactúa con el exterior, mediante dispositivos de Entrada/Salida (Input/Output = I/O)
 - La CPU puede tener varias unidades internas



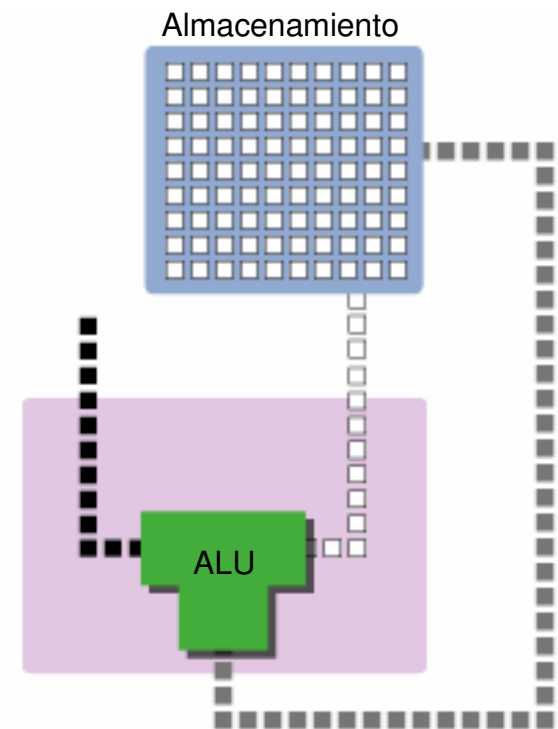
Johann von Neumann
(1903-1957)



Microprocesadores

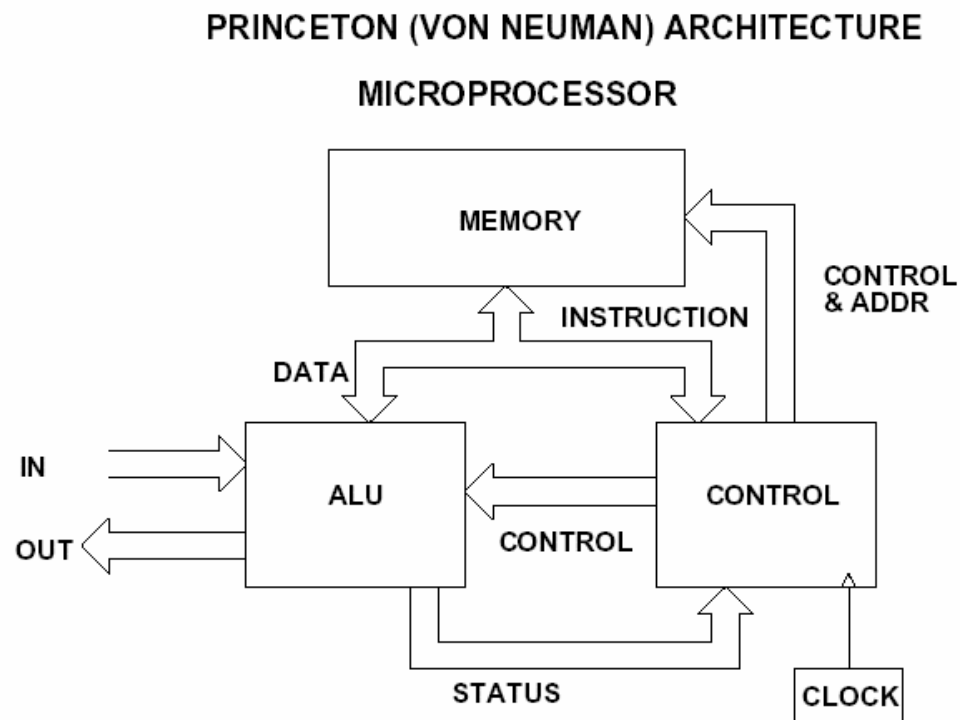
- A nivel funcional
 - Los microprocesadores, toman
 - Instrucciones
 - Datos de entrada
 - Y entregan
 - Datos de salida (Resultados)

$$\boxed{2} \quad \boxed{+} \quad \boxed{3} \quad = \quad \boxed{5}$$



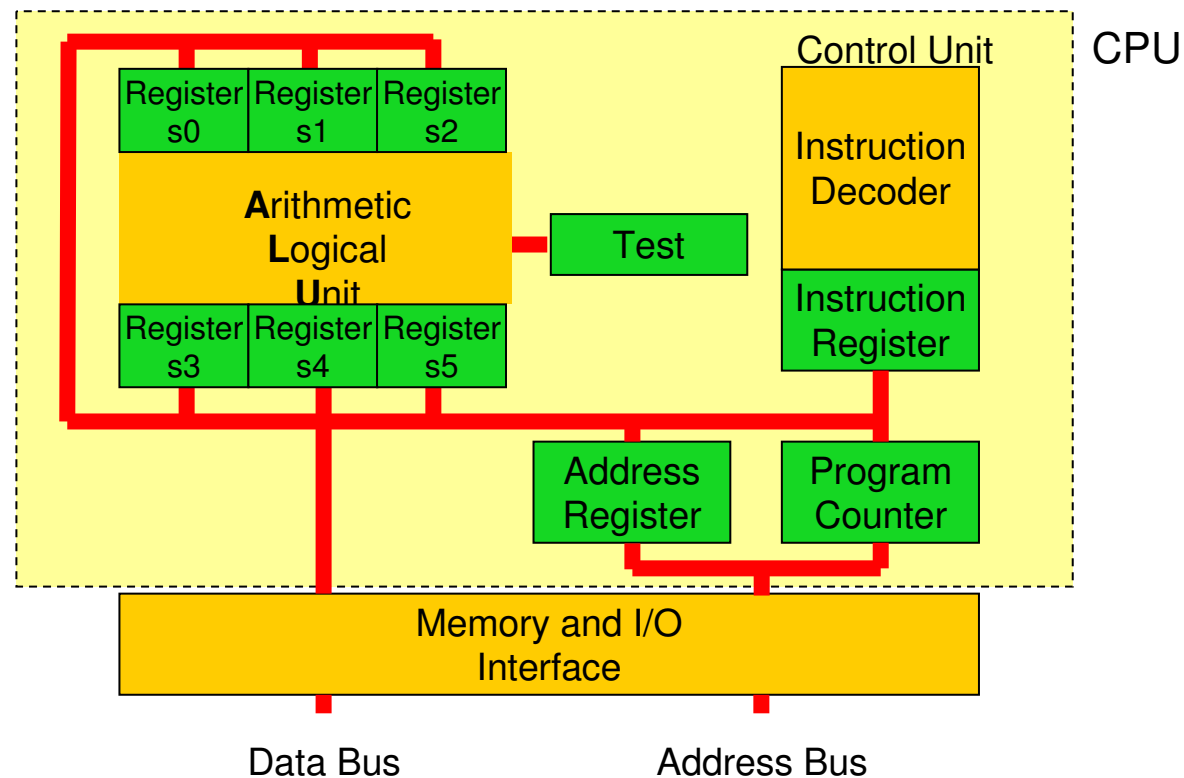
Microprocesadores

- En la práctica
 - La CPU esta compuesta por
 - Unidad de Control (que lee, ejecuta instrucciones, y controla la ALU)
 - Unidad Aritmética (ALU), que realiza las operaciones lógicas.



Microprocesadores

- Microprocesadores (CPU)
 - La arquitectura básica de un computador real, contiene una CPU, Memoria y Dispositivos de entrada/salida.
 - La CPU es lo que se implementa por medio de Microprocesadores.





Microprocesadores

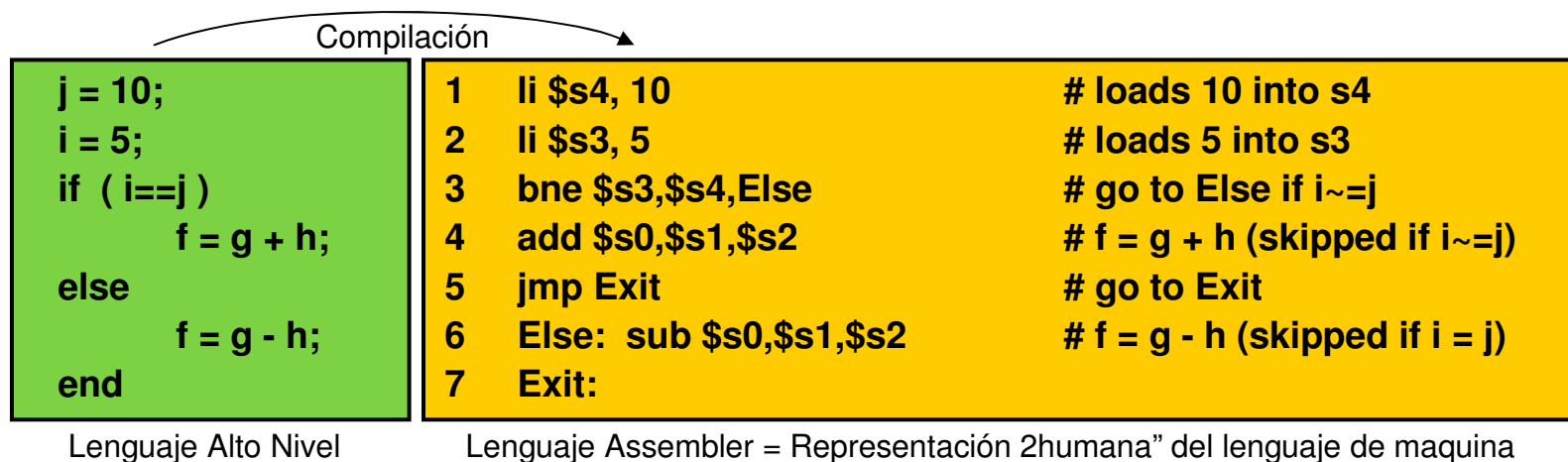
- Elementos Básicos de un Microprocesador
 - **Registros:** Almacenan información temporal. Suelen ser de 8, 16, 32 o 64 bits.
 - Propósito General: Para ser usado en operaciones.
 - Address Counters: Contienen un “puntero” a una posición de memoria que puede ser referenciada.
 - Program Counter (PC): Indica la posición de memoria que contiene la siguiente instrucción a ejecutarse.
 - **ALU:** *Arithmetic Logical Unit*, realiza operaciones aritmético/lógicas, sobre los datos de los registros
 - **Control Unit:** Decodifica las instrucciones, y las ejecuta sobre la ALU.



Microprocesadores

- Programas

- Los programas son un conjunto de instrucciones, que son extraídas de memoria, para ser ejecutadas.
- Las instrucciones son codificadas en “Lenguaje de Maquina”, que son comprensibles por el Microprocesador.
- Los programas escritos en lenguajes de alto nivel, son llevados a “Lenguaje de Maquina”





Microprocesadores

- Programas

- Assembler es solo una representación en texto, de lo que será llevado a Lenguaje de Maquina.
 - Las instrucciones se representan por mnemónicos.
- Las instrucciones en lenguaje de Maquina, son secuencias binarias, que describen la operación a ser realizada.

1	li \$s4, 10	# loads 10 into s4
2	li \$s3, 5	# loads 5 into s3
3	bne \$s3,\$s4,Else	# go to Else if i~j
4	add \$s0,\$s1,\$s2	# f = g + h (skipped if i~j)
5	jmp Exit	# go to Exit
6	Else: sub \$s0,\$s1,\$s2	# f = g - h (skipped if i = j)
7	Exit:	

add \$s0,\$s1,\$s2 se traduce a:
000000 10001 10010 10000 00000 10000
op s1 s2 s0 Shift add
amt

Luego 00000010001100101000000000100000 representa la instrucción f = g + h



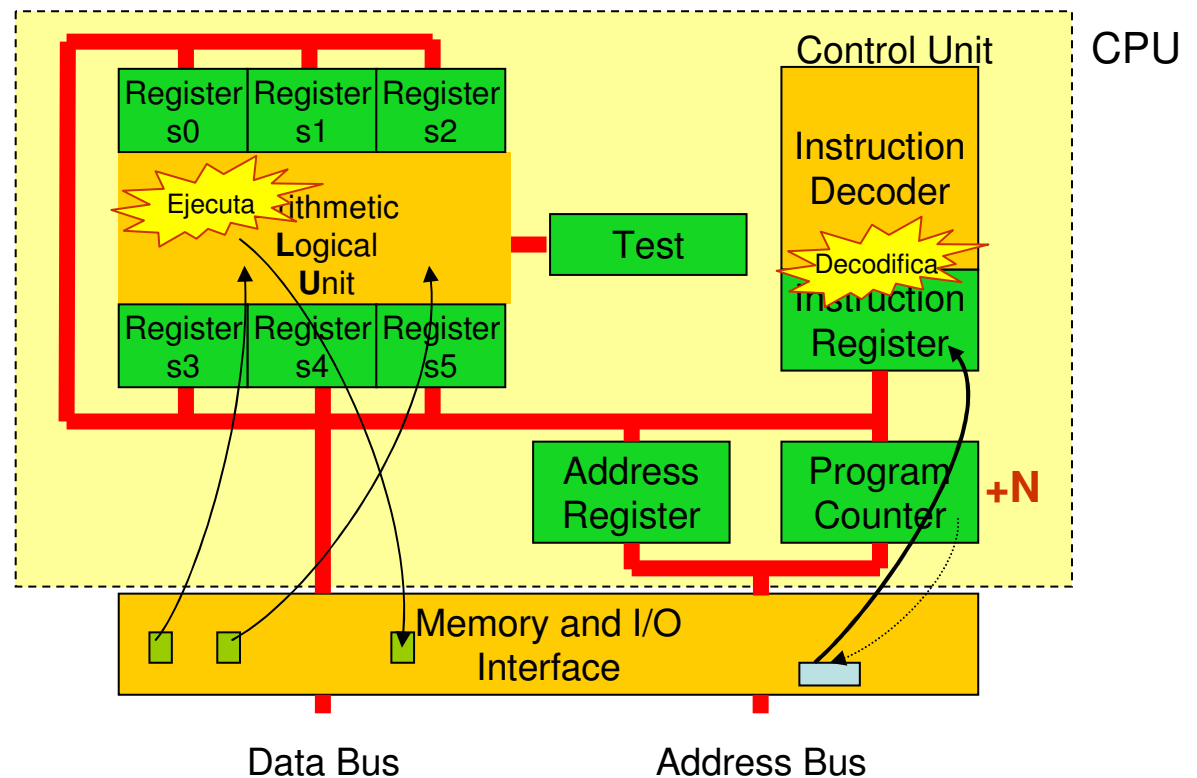
Microprocesadores

- El Clock
 - Todos los estados del microprocesador se desarrollan en tiempo discreto.
 - Se utilizan fuentes de tiempo (clock) que definen la temporización para los cambios de estado.
 - Todos los procesos al interior del Microprocesador se desarrollan en uno o más “ciclos de CPU”.
 - Distintos microprocesadores, tienen distintos rendimientos, por “cada ciclo de CPU”

Microprocesadores

• Ciclo de Ejecución de Instrucciones

1. Extrae la instrucción de la posición de memoria apuntada por el PC, y la lleva al IR (*Instruction Register*)
2. Incrementa el PC (para que apunte a la siguiente instrucción)
3. Decodifica Instrucción
4. Extrae datos de Memoria, (si son requeridos por la instrucción)
5. Ejecuta la instrucción
6. Almacena los resultados en el lugar adecuado (registros, memorias)
7. Va a 1.





Microprocesadores

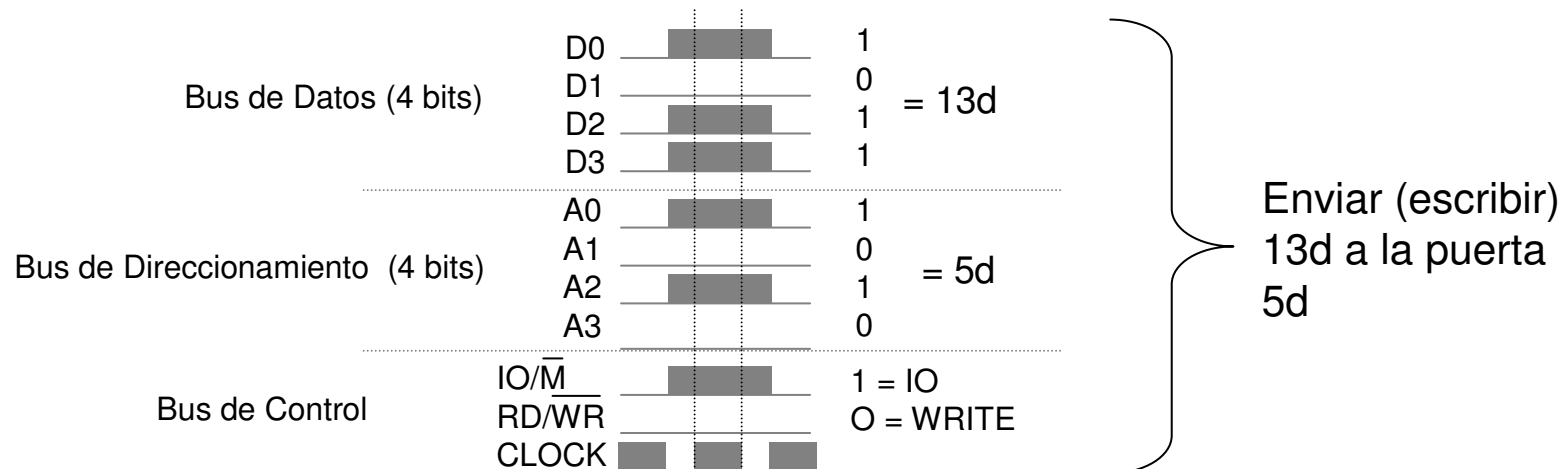
- “Largo de Palabra”
 - Se define como “palabra” (word) a la unidad de información que es transferida desde/hacia memoria, o entre registros.
 - Los primeros microprocesadores eran de 8 o 16 bits.
 - Los actuales, usan 32 bits.
 - La nueva generación, alcanzan 64 bits.
 - Largos de palabra mayores, permiten:
 - Transferir más información por ciclo de reloj
 - Realizar operaciones más complejas (operaciones lógicas o aritméticas de mayor precisión) en menos tiempo.
 - Multiplicar dos números de 32 bits, requiere
 - » 16 multiplicaciones (si palabra=8 bits)
 - » 4 multiplicaciones (si palabra=16 bits)
 - » 1 multiplicación (si palabra=32 bits)

Microprocesadores

• Direcccionamiento

– Los microprocesadores interactúan con la memoria y/o dispositivos E/S, mediante 3 buses,

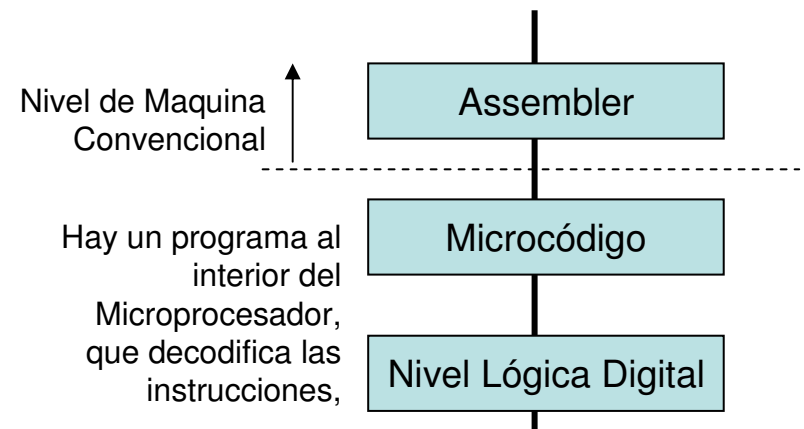
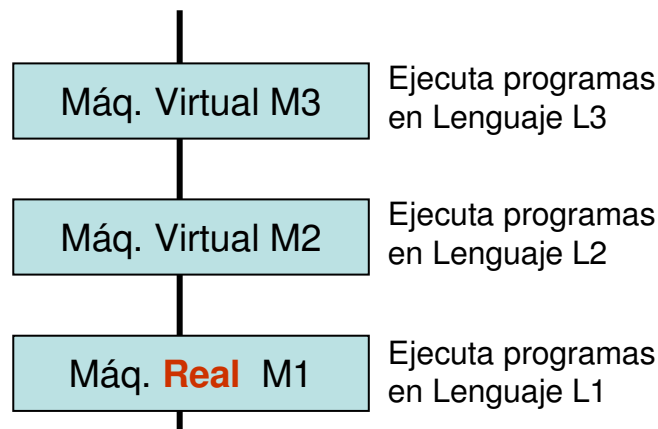
- Bus de Direcccionamiento: Indica la dirección de memoria (o dispositivo E/S) sobre la que se quiere actuar (Leer o Escribir)
- Bus de Datos: Indica los datos que se quieren leer o escribir, desde/hacia memoria o dispositivos de E/S.
- Bus de Control: Indica que se quiere hacer: Leer/Escribir, acceder memoria, o dispositivo E/S, etc..



Maquinas Virtuales

- Maquinas Virtuales

- Es un sistema que ejecuta instrucciones escritas en un determinado lenguaje de maquina.
- La maquina virtual, puede ser un programa escrito en otra maquina virtual o real.
- Los programadores de un Microprocesador, solo tienen acceso a programar a nivel de maquina convencional.



Microprocesadores

- Instrucciones Lenguaje Maquina (Assembler)

- Cada familia de microprocesadores define un “instruction set”, que permite programarlo.
- Las instrucciones realizan operaciones lógicas o aritméticas, sobre datos.
- Los datos pueden ser constantes, estar en los registros, en memoria, o estar apuntados por los registros.

- Ej:

	Constante	Registro	
ADD AX, 0123h	→		AX=AX+123h
ADD AX, BX		→	AX=AX+BX
ADD AX, [0123h]			AX=AX + Posición 123h memoria
ADD AX, [BX]			AX=AX + Posicion apuntada por BX

Puntero a Memoria Posición de Memoria



Microprocesadores

- Tipos de Instrucciones
 - Lógicas
 - Aritméticas
 - Branching (saltos)
 - Condicionales
 - Stack
 - Otras
 - Strings
 - Lookup

(Microprocesadores)

- La Ecuación de Rendimiento
 - El tiempo de ejecución de un programa es directamente proporcional
 - El periodo de reloj del Microprocesador
 - Los ciclos de reloj que toma ejecutar las instrucciones
 - Las cantidad de instrucciones que contiene el programa

$$\frac{\text{Tiempo}}{\text{Programa}} = \frac{\text{Tiempo}}{\text{Ciclo}} \times \sum \left[\frac{\text{Ciclos}}{\text{Instruccion}} \right]_i \times \frac{\text{Instrucciones}_i}{\text{Programa}}$$

$1/f_{\text{cpu}}$
 CPI_i

- Cada tipo de instrucción, toma determinado ciclos de reloj en ejecutarse.

Ejercicio:

Procesador: Pentium III @ 300Mhz

Instrucciones:

Lógicos	2500 (6 ciclos)
Suma:	7500 (6 ciclos)
Multiplicación:	2500 (12 ciclos)

¿Cuánto tarda el programa?

Microprocesadores

- En general, las aplicaciones (programas) que se escriben, realizan operaciones muy simples

Tipo Instrucción	Fortran	C	Pascal	Promedio	Términos	%
Asignación	51 %	48 %	45 %	48 %	0	0 %
Condiciones	10 %	43 %	29 %	27 %	1	80%
Llamadas a Procedimientos	5 %	12 %	15 %	11 %	2	15%
Iteraciones	9 %	3 %	5 %	6 %	3	3%
Goto	9 %	3 %	0 %	4 %	4	2%
Otras	16 %	1 %	6 %	8 %	5	0%

- El 86% son Asignaciones, Condiciones o Llamadas a Procedimientos
 - El 95% ocupa 1 o 2 parámetros.
- Las instrucciones “especiales” le agregan mucha complejidad al Microprocesador
 - Se debe implementar microcódigo para ejecutar estas instrucciones.



RISC & CISC

- RISC (**R**educed **I**nstruction **S**et **C**omputers)
 - En los 70s se pensó en ocuparse solamente del 85-90% de las instrucciones mas simples, para producir microprocesadores más simples.
 - Las instrucciones especiales, deben ser transformada por el programador, a instrucciones simples.

CISC

Extenso conjunto de Instrucciones, de modo de poder compilar en forma sencilla los programas de alto nivel, a Lenguaje de Maquina.
El microcódigo debe decodificar las instrucciones, y lograr su ejecución

RISC

Reducido conjunto de Instrucciones, de modo de hace un microprocesador mas simple, rápido y barato..
No hay microcódigo, ya que las instrucciones son muy simples.
El compilador debe llevar el programa a instrucciones simples.

RISC & CISC

- Una comparación de ambas arquitecturas.

RISC	CISC
Instrucciones sencillas en 1 ciclo	Instrucciones complejas en varios ciclos
Solo LOAD/STORE acceden a memoria.	Cualquier instrucción puede referenciar a memoria
Instrucciones ejecutadas por hardware (lógica digital)	Instrucciones ejecutadas por Microcódigo
Instrucciones de formato fijo	Instrucciones de formato variable
Pocas instrucciones y modos	Muchas instrucciones y modos
La complejidad esta en el compilador	La complejidad esta en el Microcódigo
Varios conjuntos de registros	Un solo conjunto de registros.



RISC & CISC

- Sumar una variable a otra.

a += b;

- **a** en posición de memoria 12h (hexa)
- **b** en posición de memoria 13h (hexa)

- Ejemplo CISC

MOV BX, [12h] 8 Ciclos

ADD [13h], BX 16 Ciclos

24 Ciclos

- Ejemplo RISC

LOAD A, #12

LOAD B, #13

ADD A, B

STORE A, #13

4 Ciclos

Ejemplo ficticio



Microprocesadores

- Los tiempos de ejecución de una instrucción dependen de los modos de direccionamiento, y de la generación del procesador
- Ejemplo, Instrucción ADD, en distintas generaciones de procesadores Intel.

ADD - Arithmetic Addition

Usage: ADD dest,src

Modifies flags: AF CF OF PF SF ZF

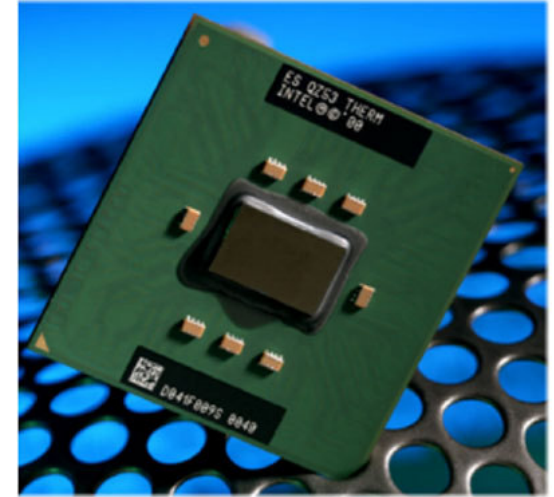
Adds "src" to "dest" and replacing the original contents of "dest".

Both operands are binary.

Operands		Clocks			Size	
		808x	286	386	486	Bytes
reg,reg	3	2	2	1	2	
mem,reg	16+EA	7	7	3	2-4	(W88=24+EA)
reg,mem	9+EA	7	6	2	2-4	(W88=13+EA)
reg,immed	4	3	2	1	3-4	
mem,immed	17+EA	7	7	3	3-6	(W88=23+EA)
accum,immed	4	3	2	1	2-3	

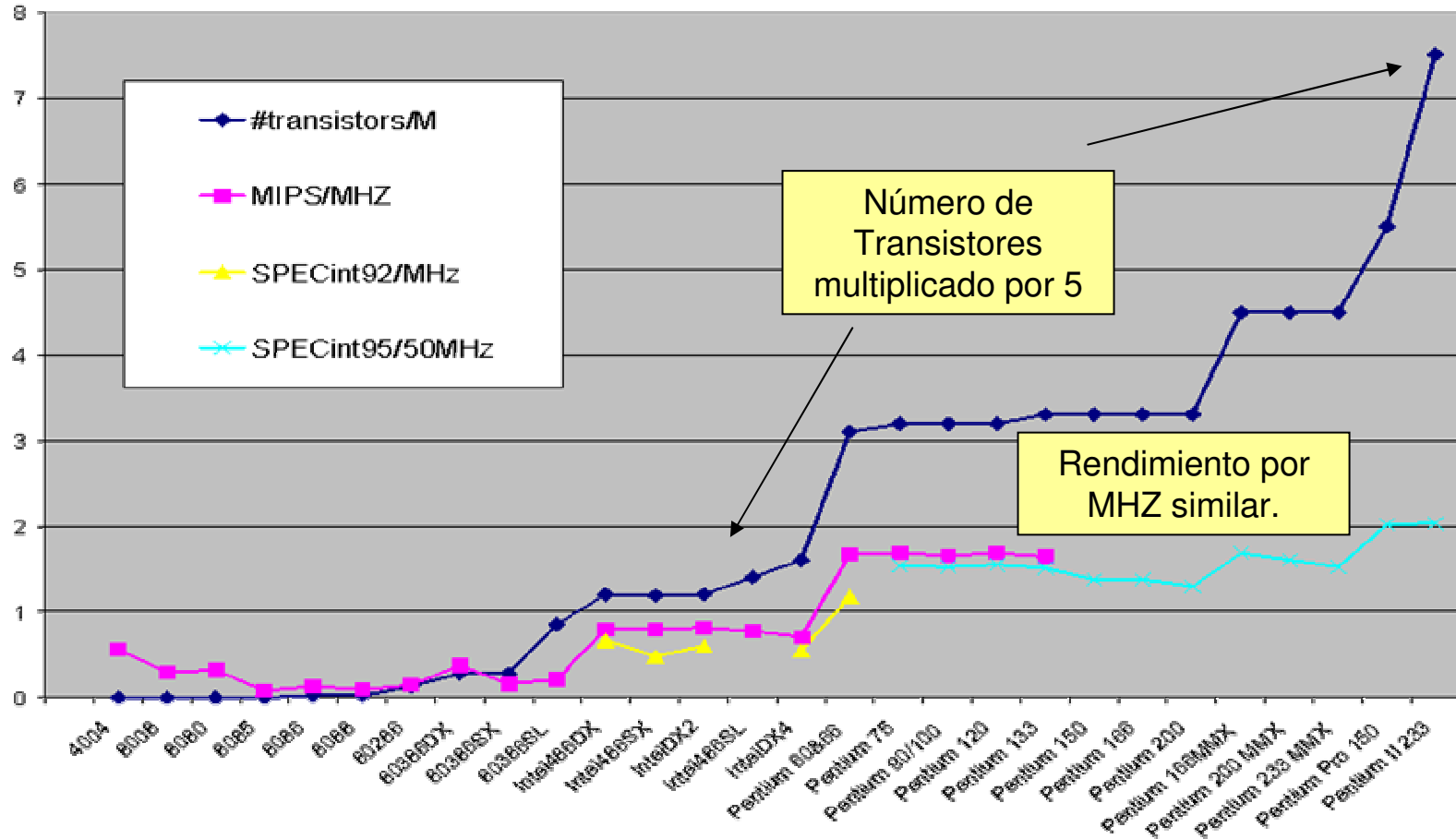
Arquitectura INTEL

- La Arquitectura de microprocesadores de mayor éxito comercial
 - 1978: Intel 8086
 - 29k Transistores
 - 16bits datos, 20bits, direccionamiento
 - 5MHz with 0.33 MIPS
 - 1982: Intel 80286
 - 134k transistores
 - 16bits datos, 24bits, direccionamiento
 - 8MHz, 10MHz with 1.5 MIPS
 - 1985: Intel 80386
 - 127k transistores
 - 32 bits
 - 16MHz with 5 to 6 MIPS
 - 1989: Intel 80486
 - 1.2M transistores
 - 25MHz with 20 MIPS
 - 1993: Pentium
 - 60MHz with 100 MIPS
 - 1999: Pentium III
 - 9.5 M Transistores
 - 2001 Pentium 4
 - 2400Mhz with 6500MIPS
 - 3200Mhz with 10000MIPS



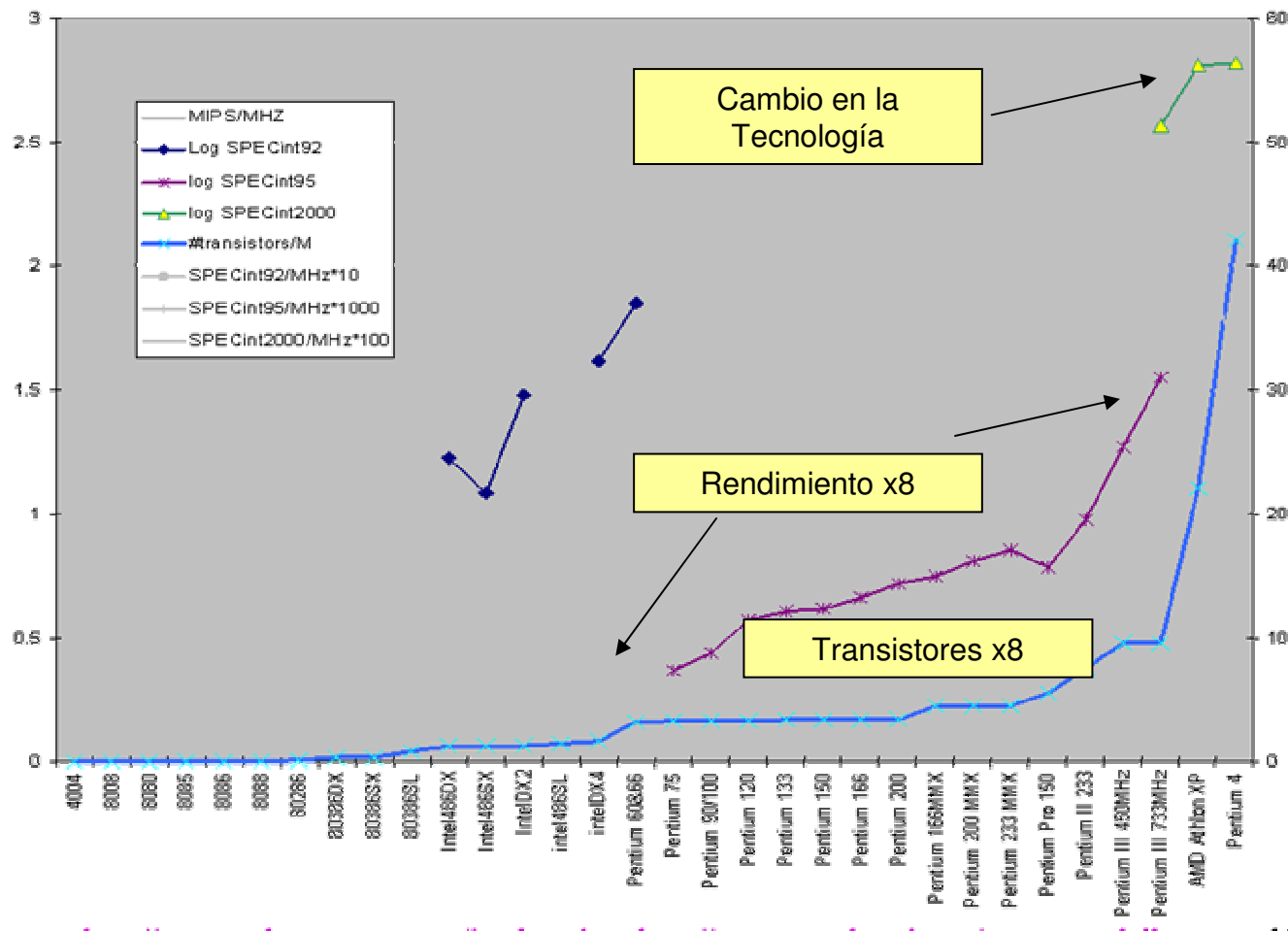
Arquitectura INTEL

- El aumento en el numero de transistores, no ha logrado aumentar la eficiencia del microprocesador (/ ciclo de CPU)



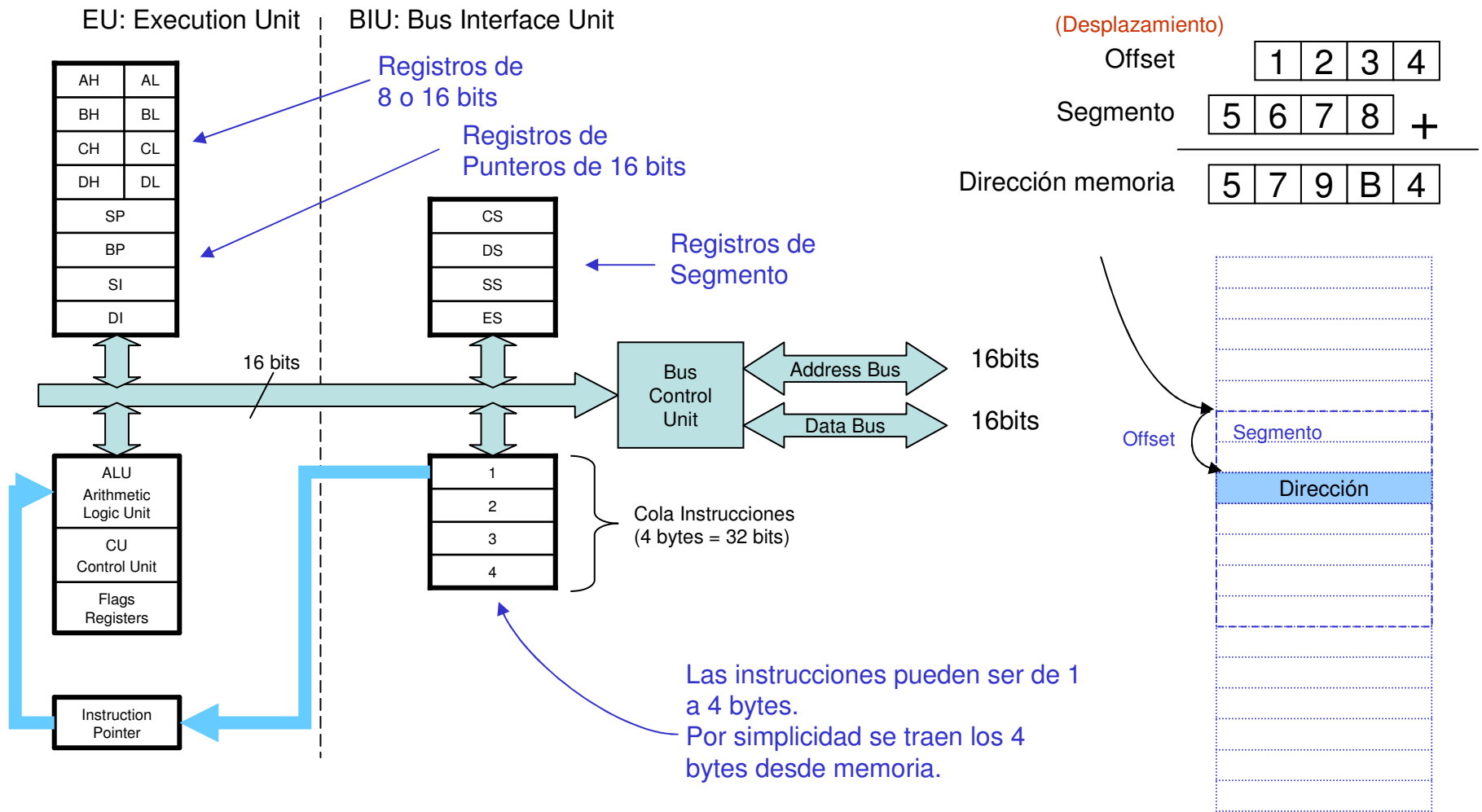
Arquitectura INTEL

- Sin embargo, el aumento en el CLOCK, hace que el rendimiento alcance el crecimiento en la densidad de transistores.



Arquitectura INTEL

• Diagrama Interno y Direccionamiento



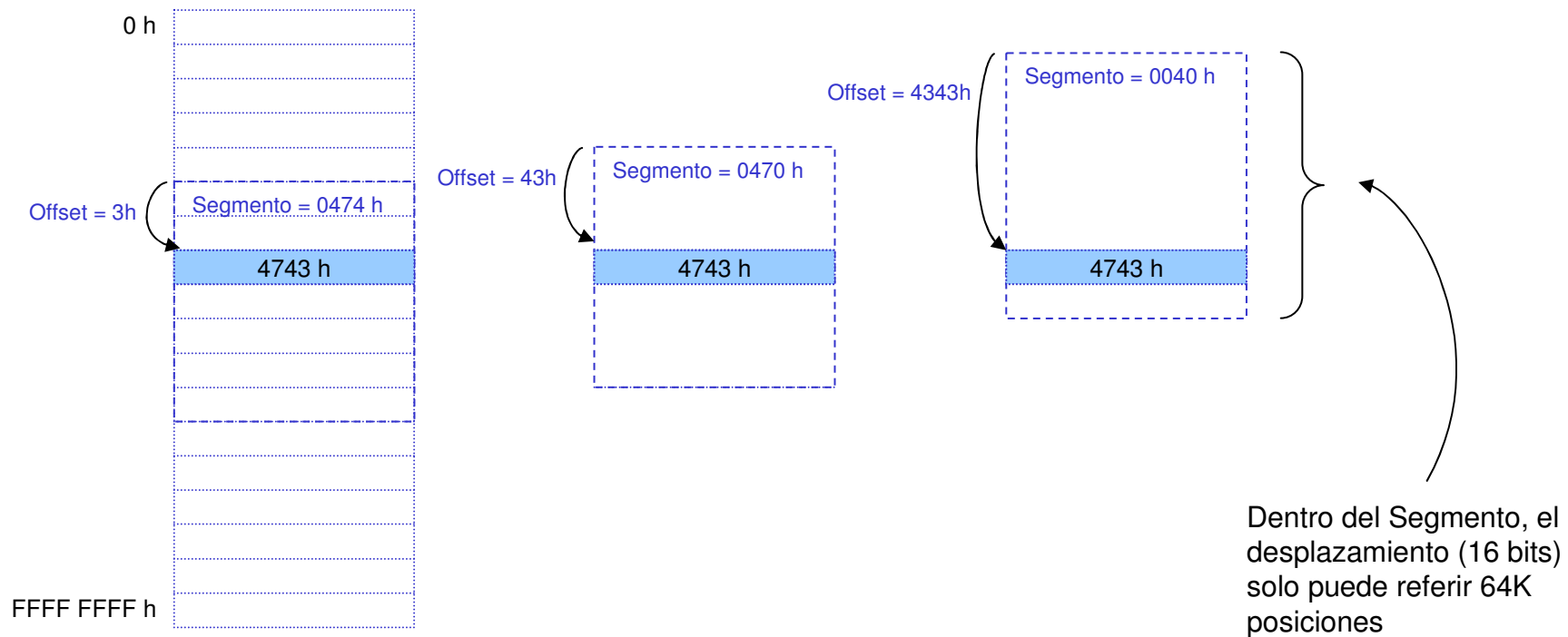


Arquitectura INTEL

- Segmentos y Desplazamiento

- Los segmentos son móviles
- Existen varias Seg:Off de referenciar una posición de memoria.

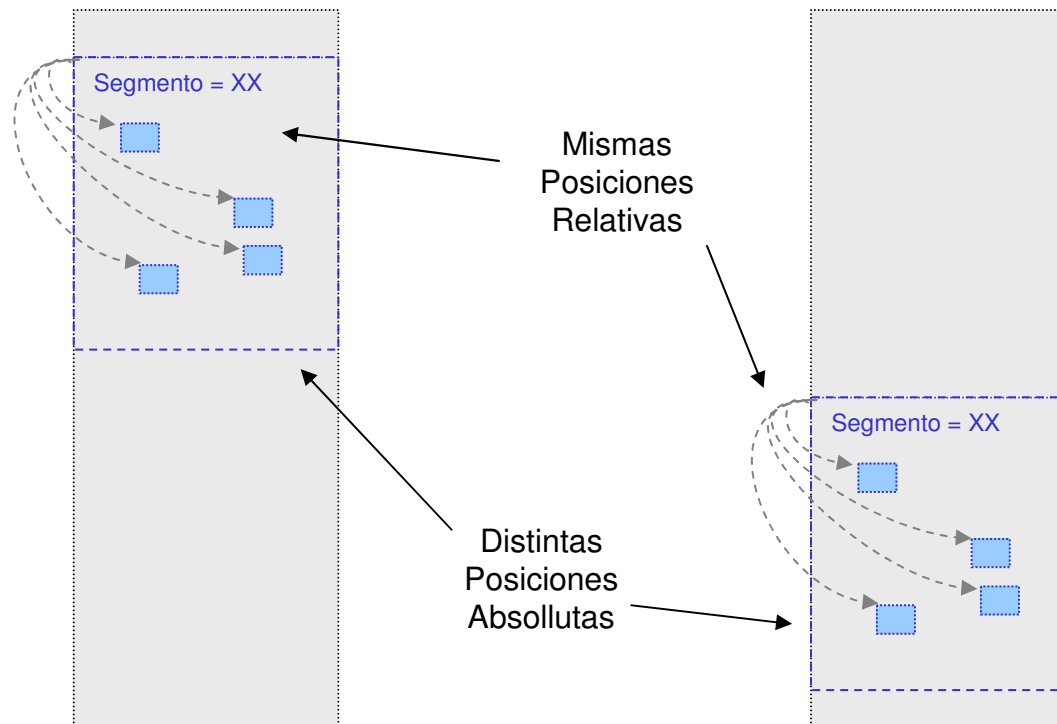
$$\text{Pos}[\text{Seg:Off}] = \text{Pos}[16\text{d} * \text{Seg} + \text{Off}] = \text{Pos}[10\text{h} * \text{Seg} + \text{Off}]$$



Arquitectura INTEL

- Programas “ReUbicables”?

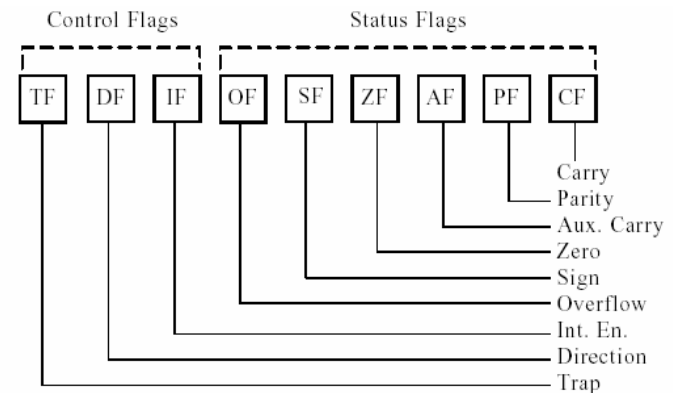
- Los Segmentos permiten que el direccionamiento de memoria sea siempre referido al origen del segmento.
- Con esto, el programa se independiza de la posición de memoria real que ocupan (al momento de ejecutarse).





Arquitectura INTEL

- Registros del 8086
 - Registros de Datos
 - AX: Accumulator
 - BX: Base
 - CX: Counter
 - DX: Data
 - Registros de Punteros e Índices
 - IP: Instruction Pointer
 - SP: Stack Pointer
 - BP: Base Pointer
 - SI: Source Index
 - DI: Destination Index
 - Registros de Segmentos
 - CS: Code Segment (se usa con IP)
 - SS: Stack Segmento (Se usan con SP y BP)
 - DS: Data Segment (se usan con DI o SI)
 - ES: Extra Segment (se usan con DI o SI)

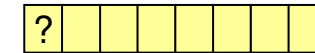


- Flags
 - Contiene el estado de la ALU, tras la ultima operación.

Flag	Verdadero (1)	Falso (0)
CF – Carry	Carry (8 bits)	
PF – Parity	Par	Impar
AF – Aux Carry	Carry (4 bits)	
ZF – Zero	Cero	No Cero
SF – Sign	Negativo	Positivo
OF – Overflow	Se produjo Rebalse	-
IF – Interrupt	Interrupciones Habilitadas	Interrupciones Deshabilitadas
DF – Direction	Decrementa DI y SI	Incrementa DI y SI
TF – Trap		

Arquitectura INTEL

• Aritmética Binaria – Con y Sin signo



Signo
0 = Positivo
1 = Negativo

	(Unsigned)	(Signed)	
11111001	249	-7	
00000010	2	+2	
① 11111011	251	-5	→ CF=0 OF=0

	(Unsigned)	(Signed)	
11111100	252	-4	
00000101	5	+5	
① 00000001	1 (no válido)	1	→ CF=1 → Falta Precisión para resultado. OF=0

	(Unsigned)	(Signed)	
01111001	121	+121	
00001011	11	+11	
① 10000100	132	-124 (no válido)	→ CF=0 OF=1 → No se pudo preservar signos

	(Unsigned)	(Signed)	
11110110	246	-10	
10001001	137	-119	
① 01111111	127 (no válido)	+127 (no válido)	→ CF=1 → Falta Precisión para resultado. OF=1 → No se pudo preservar signos

¿Carry? ¿Overflow?



Arquitectura INTEL

- Ejemplos Uso de Punteros

Siguiente Instrucción a Ejecutar	CS:[IP]
Stack	SS:[SP]
Base Pointer	SS:[BP]
Punteros de Origen Destino	DS:[SI]
	ES:[DI]

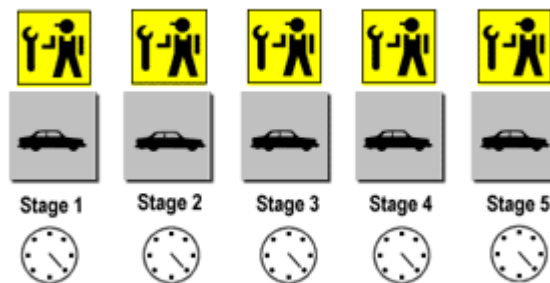


Tecnologías de alto rendimiento

- Técnicas para mejorar el rendimiento de un microprocesador
 - Pipelining
 - Branch Prediction
 - Superscalar Execution
 - Caches
 - SIMD
 - HyperThreading

Tecnologías de alto rendimiento

- Pipelining
 - Inicialmente
 - Los microprocesadores ejecutaban 1 instrucción a la vez, a través de varias etapas de ejecución.
 - Actualmente
 - Se mantienen varias etapas de ejecución, pero se tienen todas las etapas ocupadas...
 - Ciclos/Instrucción: No cambian mucho.
 - Instrucciones/Ciclo: Si cambian (cercano a 1)





Tecnologías de alto rendimiento

- Branch Prediction

- Los procesadores muchas veces deben ejecutar instrucciones condicionales, que lo pueden llevar a cambiar la secuencia de instrucciones en ejecución.
- Un ejemplo, son las instrucciones de “Salto Condicional”....

1022h CMP CX, [14h]

1025h JZ 1234h

1028h ADD CX, 234h

1234h SUB AX, BX

?

Tecnologías de alto rendimiento

• Branch Prediction

– Eso provoca ineficiencias en el Pipelining

- El Pipeline debe esperar a la instrucción condicional, antes de ingresar mas instrucciones...
- Se producen “tiempos muertos”, a la espera de ejecutar la instrucción de condicional.

Etapa Pipeline	Ciclos												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Extracción Instrucciones	1	2	B					4	5	6	7	8	9
Decodificación Instrucciones		1	2	B					4	5	6	7	8
Calculo Direcciones			1	2	B					4	5	6	7
Extracción de Operandos				1	2	B					4	5	6
Ejecución					1	2	B					4	5

Tecnologías de alto rendimiento

- Branch Prediction

- ¿Qué ocurriría si de todos modos se ingresan las instrucciones consecutivas al pipeline?

Etapa Pipeline	Ciclos						
	1	2	3	4	5	6	7
Extracción Instrucciones	1	2	B	4	5	6	7
Decodificación Instrucciones		1	2	B	4	5	6
Calculo Direcciones			1	2	B	4	5
Extracción de Operandos				1	2	B	4
Ejecución					1	2	B

Es probable que sea “trabajo perdido” en caso que efectivamente haya un salto

- **Penalización de Salto (b):**

- Tiempo perdido, al deshacer los pasos de instrucciones siguientes al salto, en caso se produzca salto hacia otro punto del programa.



Tecnologías de alto rendimiento

- Branch Prediction

- El asumir que los saltos no se realizan, para ganar tiempo en el Pipeline, tiene un impacto en caso el supuesto sea equivocado (***b***)
- Impacto en Rendimiento... sea:
 - P_J : Probabilidad que una instrucción sea de Salto Condicional
 - P_T : Probabilidad que el salto se realice
 - B : Penalización de Salto (ciclos)

Luego los “Ciclos/Instrucción Reales” son:

$$CPI' = \overbrace{(1 - P_J)CPI}^{\text{Instrucciones normales}} + P_J \left[\overbrace{P_T(CPI + b)}^{\text{Saltos que se ejecutan}} + \overbrace{(1 - P_T)CPI}^{\text{Saltos que no se ejecutan}} \right]$$

$$CPI' = CPI + P_J P_T b$$

- Si: $CPI = 1$; $P_J = 0.3$; $P_T = 0.65$ y $b=4$ se tiene que $CPI' = 1.78$
 - El rendimiento es un 56% del rendimiento potencial!!!



Tecnologías de alto rendimiento

- Branch Prediction

- La “Predicción de Rama” es una técnica que intenta “predecir” cual será la decisión que tomarán las instrucciones condicionales.
- En base a la predicción, se ingresan al Pipeline las instrucciones más probables de ser ejecutadas
 - Si la predicción es correcta: Se gana tiempo!!
 - Si la predicción es incorrecta: Se vacía el Pipeline, se vuelve al estado anterior....**Penalización del Salto (b)**
- Se mejora el comportamiento anterior si:
 - P_T se reemplaza por P_W (Probabilidad de error en la predicción)

$$CPI' = CPI + P_J P_W b$$
 - $P_W < P_T$

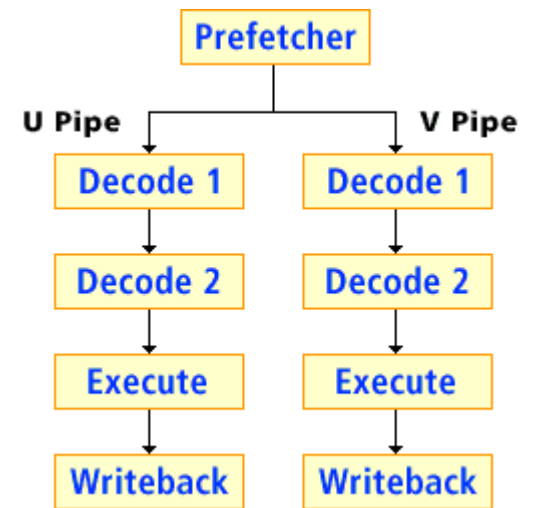


Tecnologías de alto rendimiento

- Branch Prediction
 - *Static Branch Prediction*: Se basa en reglas “predefinidas” para cada instrucción.
 - *Dynamic Branch Prediction*: Mantiene estadísticas de comportamiento para las instrucciones, y las aplica en las predicciones.

Tecnologías de alto rendimiento

- Superscalar Execution
 - Consiste en implementar 2 o más pipelines, que pueden ejecutar instrucciones “escalares” en paralelo.
 - Por lo general, no son pipelines idénticas, sino que pueden tener especialización diferente.
 - Las pipelines comparten el mismo set de registros.





Tecnologías de alto rendimiento

- Superscalar Execution

- Problema:

- Que ocurre con

- 15 ADD AX, BX

- 16 ADD AX, DX



Se ejecutan al mismo
tiempo?

- El procesador no puede operar en forma superescalar....
 - El programa debe estar construido de tal forma, de minimizar la dependencia entre instrucciones consecutivas..
 - **ILP: instruction-level parallelism**

Tecnologías de alto rendimiento

- *Memorias Cache*

- Las memorias convencionales son más lentas que la CPU
 - 1 Ciclo de CPU ~ 1ns
 - Tiempo Acceso Memoria ~ 10-60ns
- Se hace necesario mejorar el rendimiento de los accesos a memoria, a costo razonable.
- **Principio de Localidad**
 - Cuando un programa accede a memoria, es altamente probable el siguiente acceso a sea en una zona de memoria cercana.

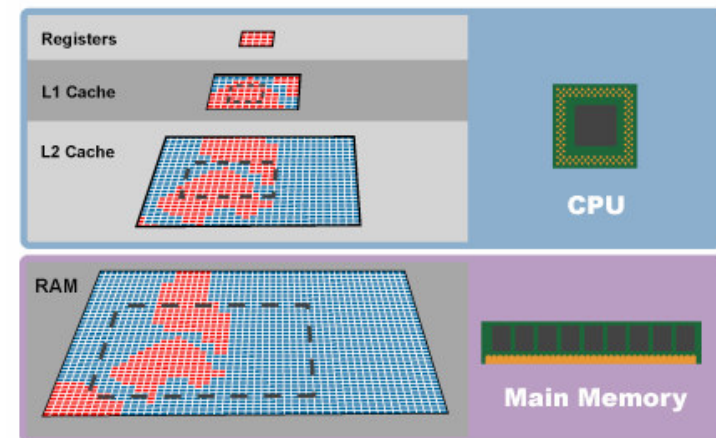
Level	Access Time	Typical Size	Technology	Managed By
Registers	1-3 ns	1 KB	Custom CMOS	Compiler
Level 1 Cache (on-chip)	2-8 ns	8 KB-128 KB	SRAM	Hardware
Level 2 Cache (off-chip)	5-12 ns	0.5 MB - 8 MB	SRAM	Hardware
Main Memory	10-60 ns	64 MB - 1 GB	DRAM	Operating System
Hard Disk	3,000,000 - 10,000,000 ns	20 - 100 GB	Magnetic	Operating System/User

Tecnologías de alto rendimiento

• Memorias Cache

- Son espacios de memoria de alto rendimiento, que se usan para acceder a los datos más frecuentes.
- Si la CPU requiere buscar un dato en memoria:
 - Se revisa si esta en el Cache [**c Ciclos**]
 - Si esta en cache (**Acierto!**): Se utiliza en forma inmediata
 - Si no esta (**Falla!**): Se traen los datos de memoria [**m Ciclos**]
- Si un dato se referencia k veces
 - 1 vez se ira a memoria [**m Ciclos**]
 - $k-1$ veces se irá al cache [**c Ciclos**]
- Se define
 - Tasa de Aciertos: $h = (k-1)/k$
 - Tasa de Fallas: $(1-h) = 1/k$
 - Tiempo medio de acceso:

$$Ta/k = c + (1-h)m$$



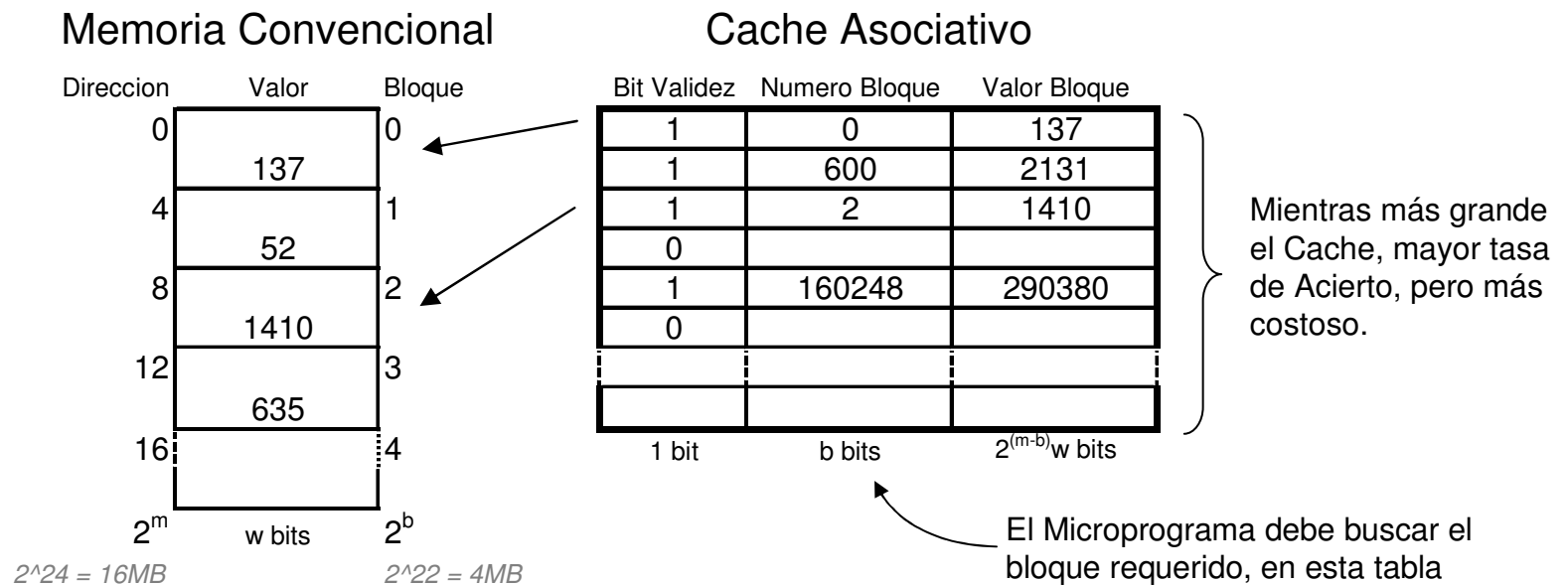


Tecnologías de alto rendimiento

- Memorias *Cache*

- *Cache Asociativos*

- *Se divide la memoria real, en bloques.*
- *Se crea una tabla (cache asociativo) que mantiene un conjunto de bloques en forma local:*
 - **Bit Validez:** Indica si “línea” esta ocupada
 - **Numero Bloque:** Indica el bloque “en cache”
 - **Valor Bloque:** Contiene los datos del bloque



Tecnologías de alto rendimiento

- Memorias *Cache*

- Caches de Mapeo Directo*

- En lugar de buscar el bloque requerido, en toda la tabla, se usan posiciones predeterminadas en el caché, para un conjunto de bloques. (Hashing)

- Por ejemplo

$$\text{Linea_Cache} = \text{Numero_Bloque} \text{ MOD } \text{Largo_Tabla}$$

$$\text{Numero_Bloque} = \lfloor \text{Dirección} / 4 \rfloor$$

	Bit Validez	Etiqueta	Valor Bloque
0	1	2	12130
	1	1	170
	1	3	2142
	0		
	0		
	0		
1023			

- Así,

Si se tienen 1024 líneas en la tabla de Cache, y

Se quiere ir a buscar posición 32.774 de memoria (bloque 8193)

Ésta estará en **Linea_Cache = 8193 mod 1024 = 1**

Tecnologías de alto rendimiento

- **Memorias Cache**

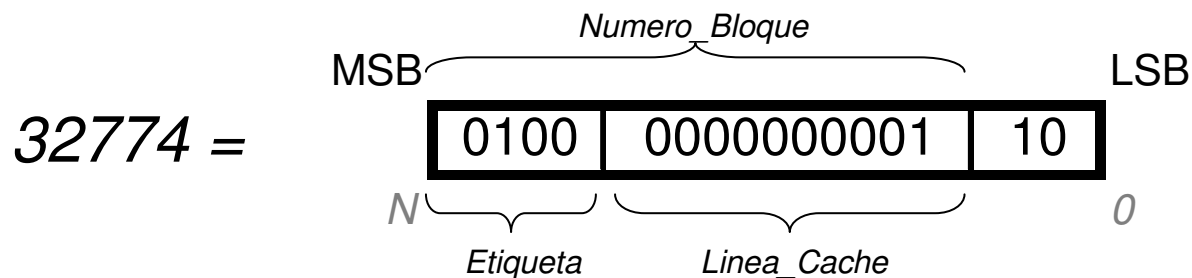
- *Caches de Mapeo Directo*

- *De igual forma, la posición 40.965, también queda en la misma Línea_Cache..*

Posición 40.965 de memoria (bloque 10241)

$$\text{Linea_Cache} = 10241 \bmod 1024 = 1$$

- *Se requiere el campo **Etiqueta**, para indicar el bloque exacto que se esta almacenando.*
 - *En forma más directa, estas operaciones se realizan sobre los bits de la dirección de memoria*



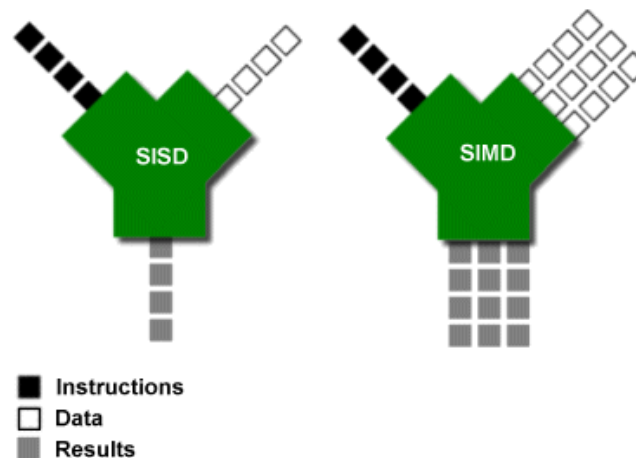


El diagrama ilustra la estructura de la memoria de la CPU, organizada en 1024 palabras, cada una de 3 bytes. Las palabras están agrupadas en tres secciones: Entrada 0 (palabras 0-255), Entrada 1 (palabras 256-511) y Entrada N (palabras 512-1023). Cada entrada contiene una columna de Validez, una de Etiqueta y una de Valor.

Tecnologías de alto rendimiento

- SISD / SIMD

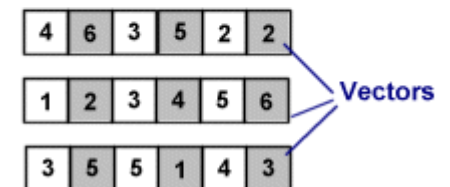
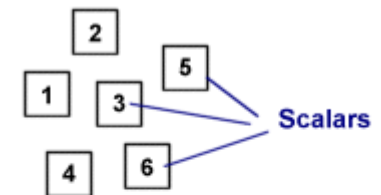
- Usualmente, se tiene un flujo de instrucciones, que procesa un flujo de datos...
- Pero (*para cierto tipo de instrucciones y cierto tipo de datos*) se pueden tener un flujo de instrucciones operando sobre varios flujos de datos.
 - SISD: Single Instruction – Single Data
 - SIMD: Single Instruction – Multiple Data



Tecnologías de alto rendimiento

- **SISD / SIMD**

- En tal caso, en lugar de escalares, se tienen vectores de datos....
- Las instrucciones operan sobre vectores (o conjuntos de escalares)
- Caso:
 - Pentium II introducía extensiones MMX
 - 57 nuevas instrucciones, con el objetivo de eficiente el procesamiento multimedia.



Tecnologías de alto rendimiento

- SISD / SIMD

Caso: Pentium MMX (SIMD)

- Tecnología diseñada para lograr óptimo rendimiento en operaciones multimedia.
- Permite realizar operaciones sobre un “vector” de datos.
- Define 4 tipos de datos “enteros” agrupados en registros de 64 bits.

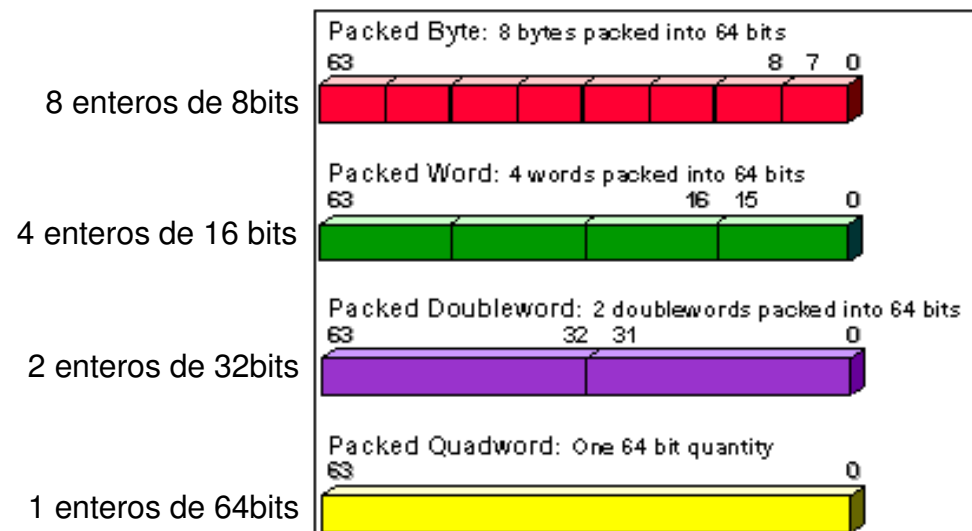
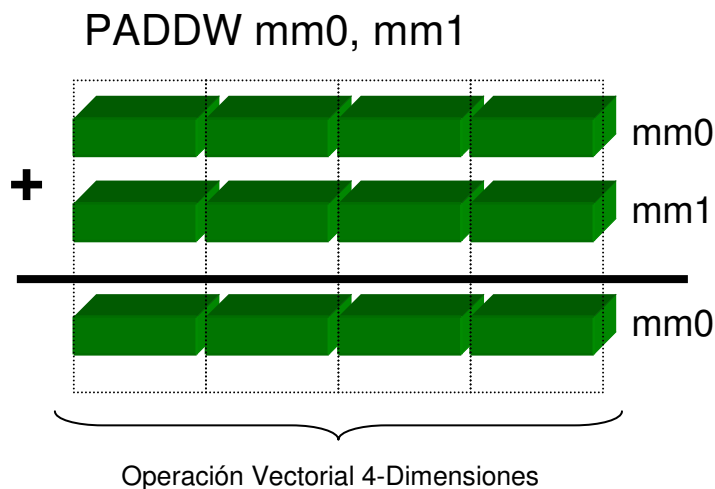


Figure 3. MMX Technology Packed Data Types

Tecnologías de alto rendimiento

- SISD / SIMD

Caso: Pentium MMX (SIMD)

- Se agregaron 57 nuevas instrucciones que realizan operaciones aritméticas directamente sobre estos “data-types”.
 - Ops. de suma = 1 ciclo
 - Ops. de Multiplicación = 3 ciclos.
- Se utilizan 8 nuevos registros de 64 bits, que eran los destinados a operaciones de punto flotante.

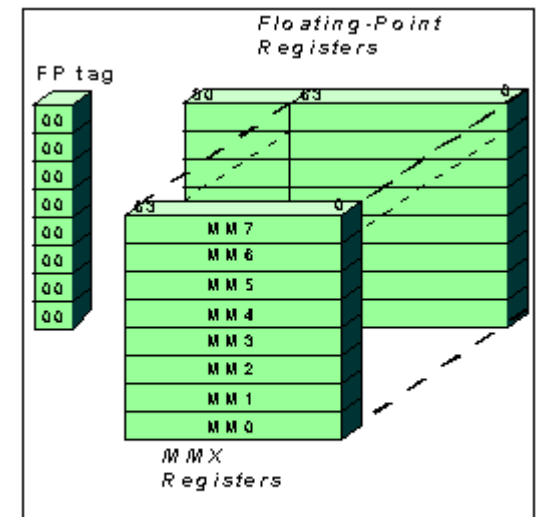
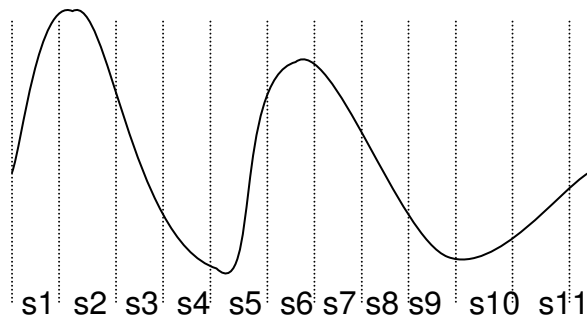


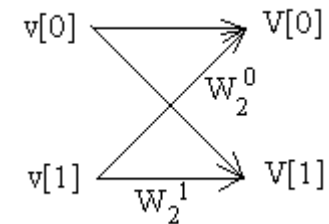
Figure 4. Mapping of MMX Registers to Floating-Point Registers

Tecnologías de alto rendimiento

- Caso: Pentium MMX (SIMD)
Aplicación: Procesamiento Señales de Audio



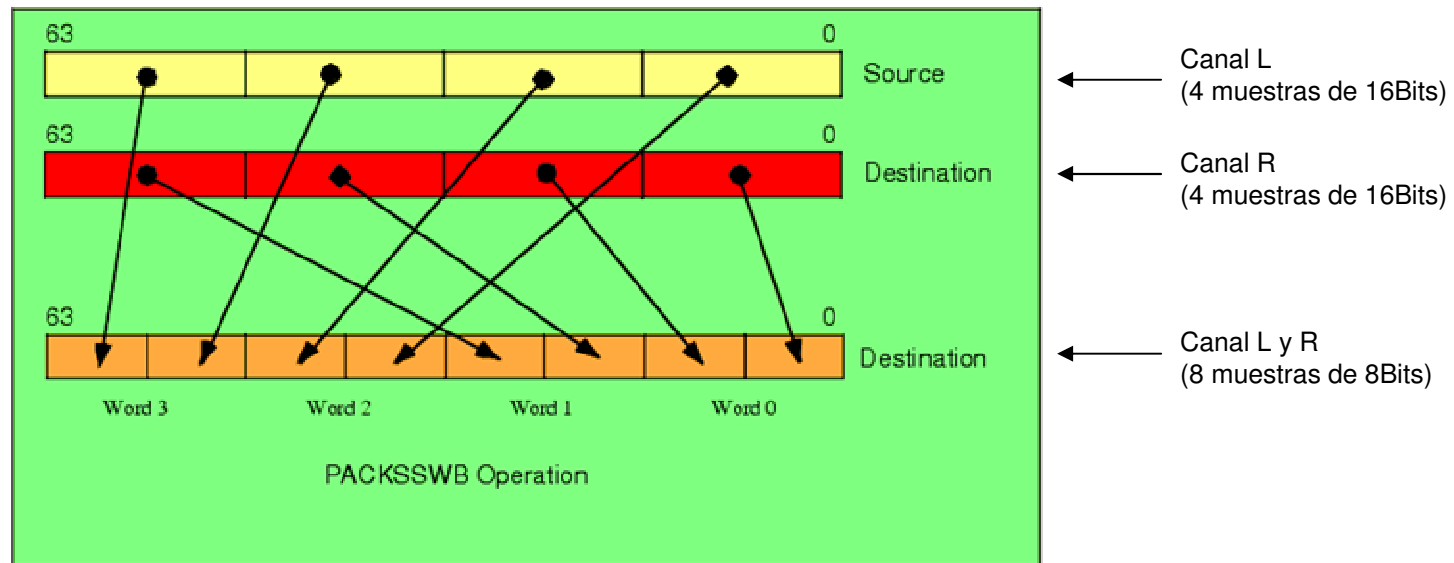
- Combinar Señales
Requiere sumas masivas sobre las muestras de las distintas señales
- Procesamiento Señales Multi-Canal
Cada canal se procesa por separado
- Filtrar Señales
Convolución sobre las muestras
- Transformación Dominio $t \rightarrow f$
Realizar FFT sobre las muestras
- Decodificación Dolby Surround
- Compresión/Descompresión MP3
Codifica las componentes de frecuencia audibles.



- Todas estas operaciones requieren realizar operaciones sobre las señales, en tiempo real (muy rápido). MMX acelera este proceso.

Tecnologías de alto rendimiento

- Caso: Pentium MMX (SIMD)
Aplicación: Procesamiento Señales de Audio
 - convertir
un stream de audio estereo 16 bits
en
un stream de audio estereo de 8 bits





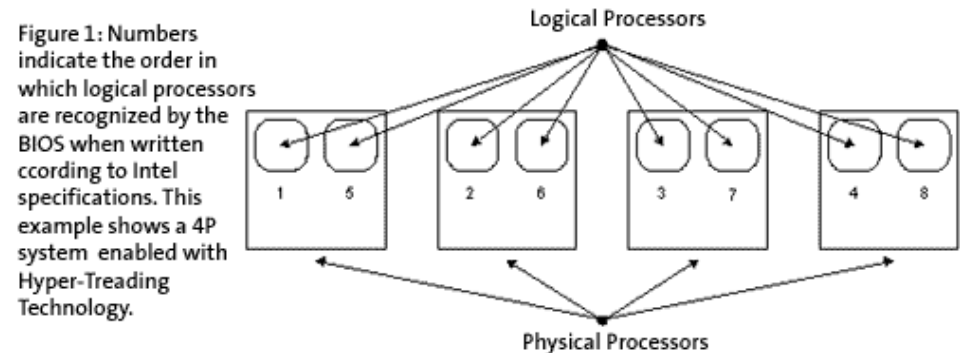
Tecnologías de alto rendimiento

- HyperThreading
 - Pentium 4 y Xeon incorporan Hyper-Threading
 - La idea es:
 - duplicar área de la CPU que mantienen información de estado (no las unidades de ejecución)
 - Permitir al Sistema Operativo enviar al procesador dos threads (procesos) independientes.
 - Al tener duplicadas las áreas de estado, el procesador tiene siempre 2 estados, entre los cuales puede conmutar.
 - Cuando un thread se detiene (acceso a memoria, fallo de cache, fallo de branch prediction, etc) se conmuta al otro estado, para permitir la ejecución del otro thread.

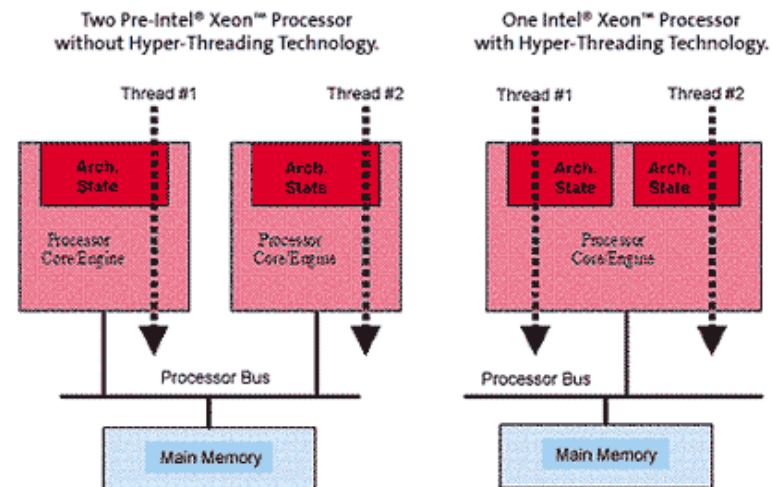
Tecnologías de alto rendimiento

- HyperThreading

- El Sistema Operativo vé 2 procesadores, y le deja la responsabilidad al Microprocesador, de cómo ejecutar ambos threads simultáneamente



- El rendimiento es mejor que 1 procesador, pero menor a 2 procesadores.





Interrupciones

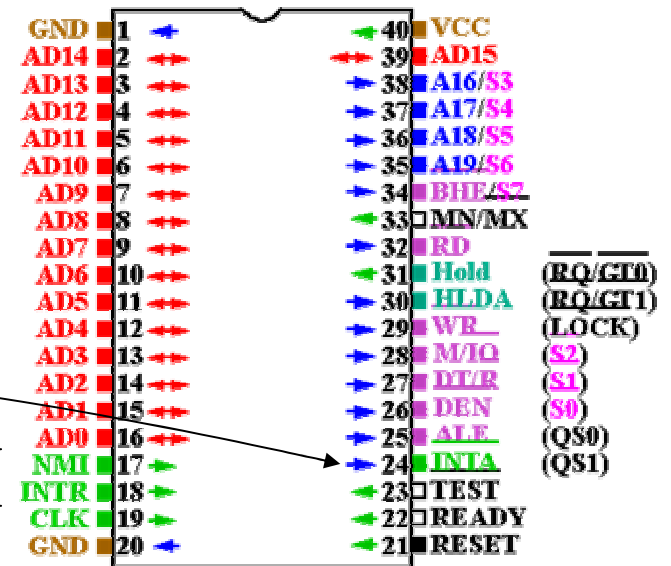
- Interrupciones

- Los microprocesadores deben ser capaces de reaccionar frente a estímulos externos.
- Para ello son capaces de reaccionar ante señales eléctricas, que se denominan interrupciones.
- Cuando se produce una interrupción, esta deben ser atendida:

- Interrupciones Enmascarables
Aquellas que solo se atienden si $IF = 1$
- Interrupciones No-Enmascarables
Aquellas que se atienden siempre

INTA – Interrupt Acknowledge
(Indica que la INT se atenderá)

NMI – Non Maskable Interrupt
INTR - INTeRrupt



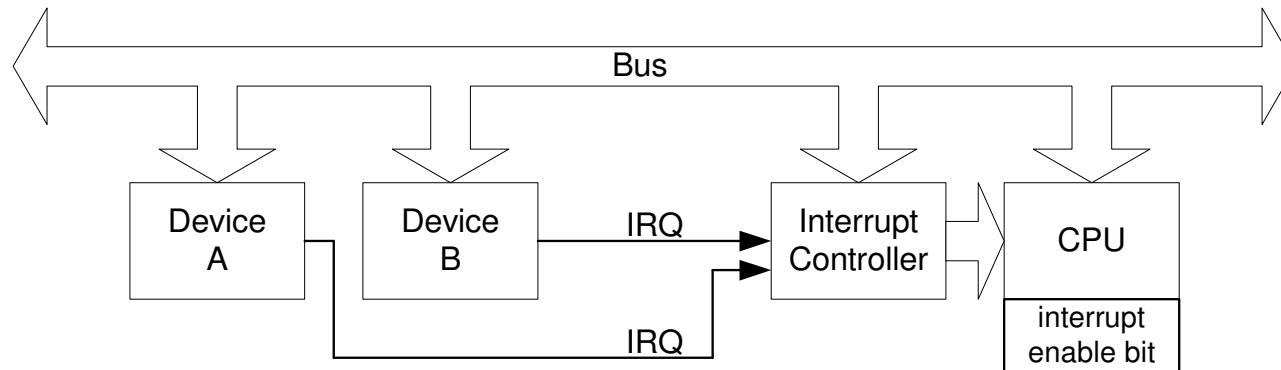


Interrupciones

- ¿Qué ocurre cuando llega una interrupción?
 - Se termina de ejecutar la instrucción en curso
 - Se almacena información de estado en el STACK
 - Se salta a una rutina de atención de interrupciones (ISR – Interrupt Service Routine)
 - Se atiende la interrupción
 - Se recupera información de estado del STACK
 - Se continua con el programa
- ¿Cómo se sabe que evento se produjo?
 - Hay varias posibles “Solicitudes de Interrupción” (IRQ – Interrupt Request) que deben ser indicadas a la CPU, para determinar la ISR a ejecutar.

Interrupciones

- Se utiliza un *Programmable Interrupt Controller*
 - Distintos dispositivos están “cableados” al P.I.C, y generan distintas IRQs.
 - Cuando se produce una IRQ, el PIC indica a la CPU el “Numero de IRQ”



- Dispositivo genera IRQ al PIC
- PIC genera un **INTR=1** (o **NMI=1**) a la CPU
- CPU responde con **INTA=0** (normalmente esta en 1)
- PIC escribe en el Bus de Datos el Interrupt Code (**nn**), y coloca **INTR=0** (o **NMI=0**)
- CPU ejecuta un **INT nn**, que ejecuta las ISR correspondiente.



Lecturas

- Branch prediction in the Pentium family
<http://www.x86.org/articles/branch/branchprediction.htm>
- MMX™ Technology Architecture Overview
http://www.intel.com/technology/itj/q31997/articles/art_2.htm
- Introduction to the Streaming SIMD Extensions in the Pentium III
http://www.x86.org/articles/sse_pt1/simd1.htm

